We claim:

1. A method for accelerating the running time of an application on a central processing unit (CPU) of a computer by adapting the code of the application in an application file to the hardware on which it runs, the method comprising:

identifying hotspot functions in the application to accelerate;

identifying the hardware on which the application runs;

extracting the code of said hotspot functions from the application file;

changing the code of said hotspot functions extracted from said application file to create new code; and

changing the flow of said application to go through said new code.

2. The method of claim 1, wherein said hotspot functions take most of the processing time.

3. The method of claim 1, wherein said step of identifying hotspot functions uses symbol information or debug information embedded in said application file to determine the boundaries of said functions.

4. The method of claim 1, wherein said step of identifying hotspot functions uses code patterns in said application to determine the boundaries of said hotspot functions.

5. The method of claim 1, wherein said step of identifying hotspot functions chooses all said functions to be accelerated.

6. The method of claim 1, wherein said step of identifying hotspot functions uses human guidance to choose said functions to be accelerated.

7. The method of claim 1, wherein said step of identifying hotspot functions further includes the steps of:

running the program code;

checking the usage of each function; and

analyzing usage statistics of each function for selecting functions to accelerate.

11164670.01

8. The method of claim 1, wherein said step of identifying the hardware applies tests on the CPU to identify the CPU.

9. The method of claim 1, wherein said step of identifying the hardware probes for peripheral hardware on the computer.

10. The method of claim 1, wherein said step of identifying the hardware probes for designated acceleration boards on the computer.

11. The method of claim 1, wherein said step of extracting code of said hotspot functions reads the code from said application file.

12. The method of claim 1, wherein said step of extracting the code of said hotspot functions reads the code from the memory when said application is loaded to the memory.

13. The method of claim 1, wherein said step of changing the code produces a code that activates a secondary processing device to apply optimization on said extracted code, wherein the new generated code runs faster on the identified hardware.

14. The method of claim 1, wherein said step of changing the code comprises the steps of: converting a binary code version to assembly code and optimizing the code wherein said code runs faster on the identified hardware.

15. The method of claim 1, wherein said step of changing the code comprises the steps of: converting a binary code version to assembly code, converting the assembly code to C code and optimizing the code to wherein said code runs faster on the identified hardware

16. The method of claim 1, wherein said step of changing the flow of said application changes said application file.

17. The method of claim 1, wherein said step of changing the flow of said application changes the memory after said application is loaded.

11164670.01

18. The method of claim 1, wherein said step of changing the flow of said application uses dynamically loadable modules.

19. The method of claim 1, wherein said step of changing the flow of said application links the application with said new code.

20. The method of claim 1, wherein said step of changing the flow of said application changes the code to jump to said new code.

21. The method of claim 1 wherein more than one version of changed codes is generated using different optimization parameters, and further comprises the step of selecting the best version.

22. The method of claim 23, wherein said step of selecting the best version runs the different code version and selects the fastest version.

11164670.01